Strong Call-by-Value and Multi Types

Beniamino Accattoli[1] **Giulio Guerrieri**[2] Maico Leberle[1]

[1]INRIA Saclay, France

[2]University of Sussex, UK

Kick-off meeting of the International Research Network – Logic & Interaction (IRN-LI)
Rome, Italy, 13 May 2025

Outline

1. Introduction: Call-by-Value λ-Calculi

2. Our Contributions (for Strong CbV)

Table of Contents

A specific λ-calculus among a plethora of λ-calculi

The λ-calculus is the model of computation underlying

- functional programming languages (Haskell, OCaml, LISP, . . .)
- proof assistants (Coq, Isabelle/Hol, Lean, Agda, . . .).

A specific λ-calculus among a plethora of λ-calculi

The λ-calculus is the model of computation underlying

- functional programming languages (Haskell, OCaml, LISP, ...)
- proof assistants (Coq, Isabelle/Hol, Lean, Agda, ...).

Actually, there are many λ-calculi, depending on

- the evaluation mechanism (e.g., call-by-name, call-by-value, call-by-need);
- computational feature the calculus aims to model (e.g., pure, non-determinism);
- the type system (e.g. untyped, simply typed, second order).

A specific λ-calculus among a plethora of λ-calculi

The λ-calculus is the model of computation underlying

- functional programming languages (Haskell, OCaml, LISP, ...)
- proof assistants (Coq, Isabelle/Hol, Lean, Agda, ...).

Actually, there are many λ-calculi, depending on

- the evaluation mechanism (e.g., call-by-name, call-by-value, call-by-need);
- computational feature the calculus aims to model (e.g., pure, non-determinism);
- the type system (e.g. untyped, simply typed, second order).

In this talk: pure untyped call-by-value λ-calculus (mainly).

Call-by-Name vs. Call-by-Value (for dummies)

- Call-by-Name (CbN): pass the argument to the calling function before evaluating it.
- Call-by-Value (CbV): pass the argument to the calling function after evaluating it.

Call-by-Name vs. Call-by-Value (for dummies)

- Call-by-Name (CbN): pass the argument to the calling function before evaluating it.
- Call-by-Value (CbV): pass the argument to the calling function after evaluating it.

$$(\lambda x. x + x)(2 * 3)$$

CbN:

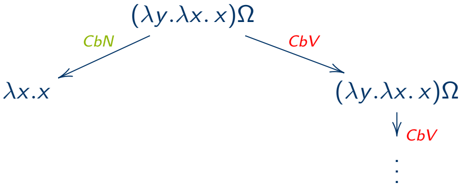
$$(\lambda x. x + x)(2 * 3) \to_{CbN} (2 * 3) + (2 * 3) \to_{CbN} 6 + (2 * 3) \to_{CbN} 6 + 6 \to_{CbN} 12$$

CbV:

$$(\lambda x. x + x)(2 * 3) \to_{CbV} (\lambda x. x + x)6 \to_{CbV} 6 + 6 \to_{CbV} 12$$

Call-by-Name vs. Call-by-Value (for dummies)

- Call-by-Name (CbN): pass the argument to the calling function before evaluating it.
- Call-by-Value (CbV): pass the argument to the calling function after evaluating it.



(Ω is a diverging program)

Summing up, CbV is eager, that is,

1. CbV is smarter than CbN when the argument must be duplicated;
2. CbV is sillier than CbN when the argument must be discarded.

Plotkin's Call-by-Value λ-calculus [Plo75]

Terms $s, t, u ::= v \mid tu$ Values $v ::= x \mid \lambda x.t$

CbV reduction $(\lambda x.t)v \to_{\beta_v} t\{v/x\}$ (restriction to β-rule)

It is closer to real implementation of most programming languages.

The semantics of CbV is completely different from standard (CbN) λ-calculus.

Plotkin's Call-by-Value λ-calculus [Plo75]

Terms $s, t, u ::= v \mid tu$ Values $v ::= x \mid \lambda x.t$

CbV reduction $(\lambda x.t)v \to_{\beta_v} t\{v/x\}$ (restriction to β-rule)

It is closer to real implementation of most programming languages.

The semantics of CbV is completely different from standard (CbN) λ-calculus.

Examples (with duplicator $\delta = \lambda z.zz$ and identity $I = \lambda z.z$):

1. $\Omega = \delta\delta \to_{\beta_v} \delta\delta \to_{\beta_v} \delta\delta \to_{\beta_v} \dots$

Plotkin's Call-by-Value λ-calculus [Plo75]

Terms $s, t, u ::= v \mid tu$ Values $v ::= x \mid \lambda x.t$

CbV reduction $(\lambda x.t)v \to_{\beta_v} t\{v/x\}$ (restriction to β-rule)

It is closer to real implementation of most programming languages.

The semantics of CbV is completely different from standard (CbN) λ-calculus.

Examples (with duplicator $\delta = \lambda z.zz$ and identity $I = \lambda z.z$):

1. $\Omega = \delta\delta \to_{\beta_v} \delta\delta \to_{\beta_v} \delta\delta \to_{\beta_v} \dots$
2. $\delta(\delta I) \to_{\beta_v} \delta(II) \to_{\beta_v} \delta I \to_{\beta_v} II \to_{\beta_v} I$ but $\delta(\delta I) \not\to_{\beta_v} (\delta I)(\delta I)$.
3. $(\lambda x.\delta)(xx)\delta$ is β_v-normal but β-divergent!
4. $(\lambda x.I)\Omega$ is β_v-divergent but β-normalizing!

A symptom that Plotkin's CbV is sick: Contextual equivalence

Def. Terms t, t' are contextually equivalent if they are observably indistinguishable, i.e., for every context $\mathtt{C}$, $\mathtt{C}\langle t\rangle \to_{\beta_v}^{*} v$ (for some value v) iff $\mathtt{C}\langle t'\rangle \to_{\beta_v}^{*} v'$ (for some value v')

A symptom that Plotkin's CbV is sick: Contextual equivalence

Def. Terms t, t' are contextually equivalent if they are observably indistinguishable, i.e., for every context $\mathtt{C}$, $\mathtt{C}\langle t \rangle \to_{\beta_v}^{*} v$ (for some value v) iff $\mathtt{C}\langle t' \rangle \to_{\beta_v}^{*} v'$ (for some value v')

Consider the terms (with $\delta \coloneqq \lambda z.zz$ as usual)

$$\omega_1 \coloneqq (\lambda x.\delta)(xx)\delta \qquad \omega_3 \coloneqq \delta((\lambda x.\delta)(xx))$$

ω_1 and ω_3 are β_v-normal but contextually equivalent to $\delta\delta$ (which is β_v-divergent)!

A symptom that Plotkin's CbV is sick: Contextual equivalence

Def. Terms t, t' are contextually equivalent if they are observably indistinguishable, i.e., for every context $\mathtt{C}$, $\mathtt{C}\langle t\rangle \to_{\beta_v}^* v$ (for some value v) iff $\mathtt{C}\langle t'\rangle \to_{\beta_v}^* v'$ (for some value v')

Consider the terms (with $\delta := \lambda z.zz$ as usual)

$$\omega_1 := (\lambda x.\delta)(xx)\delta \qquad \omega_3 := \delta((\lambda x.\delta)(xx))$$

ω_1 and ω_3 are β_v-normal but contextually equivalent to $\delta\delta$ (which is β_v-divergent)!

The "energy" (i.e. divergence) in ω_1 and ω_3 is only potential, in $\delta\delta$ is kinetic!

Why are ω_1 and ω_3 stuck? Why cannot we transform their potential energy in kinetic? It seems that in Plotkin's CbV λ-calculus something is missing...

A second symptom that Plotkin's CbV is sick: denotational semantics (1 of 2)

[Ehr12] defined a non-idempotent intersection type system for Plotkin's CbV λ-calculus.

Linear types $L ::= * \mid M \multimap N$ Multi types $M, N ::= [L_1, \ldots, L_n] \quad n \geq 0$

Idea: $[L, L', L'] \approx L \wedge L' \wedge L' \neq L \wedge L'$ (commutative, associative, non-idempotent $\wedge$).

$\rightsquigarrow$ A term $t : [L, L', L']$ can be used once as a data of type L, twice as a data of type L'.

$\rightsquigarrow$ A term $t : []$ can only be discarded during evaluation.

A second symptom that Plotkin's CbV is sick: denotational semantics (1 of 2)

[Ehr12] defined a non-idempotent intersection type system for Plotkin's CbV λ-calculus.

Linear types $L ::= * \mid M \multimap N$ Multi types $M, N ::= [L_1, \dots, L_n] \quad n \geq 0$

Idea: $[L, L', L'] \approx L \wedge L' \wedge L' \neq L \wedge L'$ (commutative, associative, non-idempotent $\wedge$).
$\leadsto$ A term $t : [L, L', L']$ can be used once as a data of type L, twice as a data of type L'.
$\leadsto$ A term $t : [\,]$ can only be discarded during evaluation.

Def: Environment Γ = function from variables to multi types s.t. $\{x \mid \Gamma(x) \neq [\,]\}$ is finite.

$$\frac{}{x : [L] \vdash x : L}\,ax \qquad \frac{\Gamma, x : M \vdash t : N}{\Gamma \vdash \lambda x.t : M \multimap N}\,\lambda \qquad \frac{\Gamma_1 \vdash v : L_1 \quad \overset{n \geq 0}{\dots} \quad \Gamma_n \vdash v : L_n}{\Gamma_1 + \dots + \Gamma_n \vdash v : [L_1, \dots, L_n]}\,! \qquad \frac{\Gamma \vdash t : [M \multimap N] \quad \Delta \vdash s : M}{\Gamma + \Delta \vdash ts : N}\,@$$

A second symptom that Plotkin's CbV is sick: denotational semantics (1 of 2)

[Ehr12] defined a non-idempotent intersection type system for Plotkin's CbV λ-calculus.

Linear types $L ::= * \mid M \multimap N$ Multi types $M, N ::= [L_1, \dots, L_n] \quad n \geq 0$

Idea: $[L, L', L'] \approx L \wedge L' \wedge L' \neq L \wedge L'$ (commutative, associative, non-idempotent $\wedge$).
$\rightsquigarrow$ A term $t : [L, L', L']$ can be used once as a data of type L, twice as a data of type L'.
$\rightsquigarrow$ A term $t : [\,]$ can only be discarded during evaluation.

Def: Environment Γ = function from variables to multi types s.t. $\{x \mid \Gamma(x) \neq [\,]\}$ is finite.

$$\frac{}{x : [L] \vdash x : L}\,ax \qquad \frac{\Gamma, x : M \vdash t : N}{\Gamma \vdash \lambda x.t : M \multimap N}\,\lambda \qquad \frac{\Gamma_1 \vdash v : L_1 \quad \overset{n \geq 0}{\dots} \quad \Gamma_n \vdash v : L_n}{\Gamma_1 + \dots + \Gamma_n \vdash v : [L_1, \dots, L_n]}\,! \qquad \frac{\Gamma \vdash t : [M \multimap N] \quad \Delta \vdash s : M}{\Gamma + \Delta \vdash ts : N}\,@$$

Rmk: The constructor for multi types (rule !) can be used only by values!
$\rightsquigarrow$ In CbV, only values can be duplicated or erased.

A second symptom that Plotkin's CbV is sick: denotational semantics (2 of 2)

Non-idempotent intersection types define a denotational model: relational semantics

$$[\![t]\!]_{\vec{x}} = \{(\Gamma, M) \mid \Gamma \vdash t : M \text{ is derivable}\} \qquad \text{where } \vec{x} \subseteq \mathsf{fv}(t)$$

Theorem (Subject reduction and expansion, [Ehr12]): If $t \to_{\beta_v} u$ then $[\![t]\!]_{\vec{x}} = [\![u]\!]_{\vec{x}}$.

A second symptom that Plotkin's CbV is sick: denotational semantics (2 of 2)

Non-idempotent intersection types define a denotational model: relational semantics

$$[\![t]\!]_{\vec{x}} = \{(\Gamma, M) \mid \Gamma \vdash t : M \text{ is derivable}\} \qquad \text{where } \vec{x} \subseteq \mathsf{fv}(t)$$

Theorem (Subject reduction and expansion, [Ehr12]): If $t \to_{\beta_v} u$ then $[\![t]\!]_{\vec{x}} = [\![u]\!]_{\vec{x}}$.

Theorem (Correctness, [Ehr12]): If $[\![t]\!]_{\vec{x}} \neq \emptyset$ then t is normalizing for "weak" β_v-reduction ("weak" = not reducing under λ's).

A second symptom that Plotkin's CbV is sick: denotational semantics (2 of 2)

Non-idempotent intersection types define a denotational model: relational semantics

$$[\![t]\!]_{\vec{x}} = \{(\Gamma, M) \mid \Gamma \vdash t : M \text{ is derivable}\} \qquad \text{where } \vec{x} \subseteq \mathsf{fv}(t)$$

Theorem (Subject reduction and expansion, [Ehr12]): If $t \to_{\beta_v} u$ then $[\![t]\!]_{\vec{x}} = [\![u]\!]_{\vec{x}}$.

Theorem (Correctness, [Ehr12]): If $[\![t]\!]_{\vec{x}} \neq \emptyset$ then t is normalizing for "weak" β_v-reduction ("weak" = not reducing under λ's).

The converse (completeness) fail!

$$[\![\omega_1]\!] = \emptyset = [\![\omega_3]\!] \quad (\text{and } [\![\delta\delta]\!] = \emptyset \text{ too!})$$

but ω_1 and ω_3 are β_v-normal, while $\delta\delta$ is β_v-divergent!

A second symptom that Plotkin's CbV is sick: denotational semantics (2 of 2)

Non-idempotent intersection types define a denotational model: relational semantics

$$[\![t]\!]_{\vec{x}} = \{(\Gamma, M) \mid \Gamma \vdash t : M \text{ is derivable}\} \qquad \text{where } \vec{x} \subseteq \mathsf{fv}(t)$$

Theorem (Subject reduction and expansion, [Ehr12]): If $t \to_{\beta_v} u$ then $[\![t]\!]_{\vec{x}} = [\![u]\!]_{\vec{x}}$.

Theorem (Correctness, [Ehr12]): If $[\![t]\!]_{\vec{x}} \neq \emptyset$ then t is normalizing for "weak" β_v-reduction ("weak" = not reducing under λ's).

The converse (completeness) fail!

$$[\![\omega_1]\!] = \emptyset = [\![\omega_3]\!] \quad (\text{and } [\![\delta\delta]\!] = \emptyset \text{ too!})$$

but ω_1 and ω_3 are β_v-normal, while $\delta\delta$ is β_v-divergent!

Rmk: Not only in relational semantics but also in other denotational models of CbV!

Summing up: a mismatch between syntax and semantics

In Plotkin's CbV λ-calculus there is a mismatch between syntax and semantics.

There are terms, such as

$$\omega_1 := (\lambda x.\delta)(xx)\delta \qquad \omega_3 := \delta((\lambda x.\delta)(xx))$$

that are β_v-normal but their semantics is the same as $\delta\delta$, which is β_v-divergent!

- semantics: context equivalence, solvability, denotational models, ...

Summing up: a mismatch between syntax and semantics

In Plotkin's CbV λ-calculus there is a mismatch between syntax and semantics.

There are terms, such as

$$\omega_1 := (\lambda x.\delta)(xx)\delta \qquad \omega_3 := \delta((\lambda x.\delta)(xx))$$

that are β_v-normal but their semantics is the same as $\delta\delta$, which is β_v-divergent!

- semantics: context equivalence, solvability, denotational models, ...

Somehow, in Plotkin's CbV λ-calculus, β_v-reduction is "not enough".

- Can we extend β_v so that ω_1 and ω_3 are divergent?
- But we want to keep a CbV discipline:

$$(\lambda x.I)(\delta\delta) \text{ is } \beta_v\text{-divergent (but } \beta\text{-normalizing)}$$

First alternative CbV λ-calculus: Fireball calculus [PaoRon99, GreLer02]

Terms $s, t ::= v \mid s\,t$

Values $v ::= x \mid \lambda x.t$

Inert terms $i ::= x \mid i\,f$

Fireballs $f ::= i \mid v$

Reduction $(\lambda x.t)f \to_{\beta_f} t\{f/x\}$ (call-by-extended-value)

First alternative CbV λ-calculus: Fireball calculus [PaoRon99, GreLer02]

Terms $s, t ::= v \mid s\,t$ Values $v ::= x \mid \lambda x.t$

Inert terms $i ::= x \mid i\,f$ Fireballs $f ::= i \mid v$

Reduction $(\lambda x.t)f \to_{\beta_f} t\{f/x\}$ (call-by-extended-value)

The fireball calculus (FC) extends β_v-reduction: ω_1 and ω_3 are β_v-normal but

$\omega_1 = (\lambda x.\delta)(xx)\delta \to_{\beta_f} \delta\delta \to_{\beta_f} \delta\delta \to_{\beta_f} \ldots$ $\omega_3 = \delta((\lambda x.\delta)(xx)) \to_{\beta_f} \delta\delta \to_{\beta_f} \delta\delta \to_{\beta_f} \ldots$

First alternative CbV λ-calculus: Fireball calculus [PaoRon99, GreLer02]

Terms $s, t ::= v \mid s\, t$ Values $v ::= x \mid \lambda x.t$

Inert terms $i ::= x \mid i\, f$ Fireballs $f ::= i \mid v$

Reduction $(\lambda x.t)f \to_{\beta_f} t\{f/x\}$ (call-by-extended-value)

The fireball calculus (FC) extends β_v-reduction: ω_1 and ω_3 are β_v-normal but

$\omega_1 = (\lambda x.\delta)(xx)\delta \to_{\beta_f} \delta\delta \to_{\beta_f} \delta\delta \to_{\beta_f} \ldots$ $\omega_3 = \delta((\lambda x.\delta)(xx)) \to_{\beta_f} \delta\delta \to_{\beta_f} \delta\delta \to_{\beta_f} \ldots$

Problem 1: No confluence: $(\lambda x.(\lambda z.z)(xx))\delta$

Problem 2: No subject reduction with multi types [Ehr12]: $(\lambda y.yy)(xx) \to_{\beta_f} (xx)(xx)$.
No subject expansion with multi types [Ehr12]: $(\lambda y.z)(xx) \to_{\beta_f} z$.

First alternative CbV λ-calculus: Fireball calculus [PaoRon99, GreLer02]

Terms $s, t ::= v \mid s\,t$ Values $v ::= x \mid \lambda x.t$

Inert terms $i ::= x \mid i\,f$ Fireballs $f ::= i \mid v$

Reduction $(\lambda x.t)f \to_{\beta_f} t\{f/x\}$ (call-by-extended-value)

The fireball calculus (FC) extends β_v-reduction: ω_1 and ω_3 are β_v-normal but

$\omega_1 = (\lambda x.\delta)(xx)\delta \to_{\beta_f} \delta\delta \to_{\beta_f} \delta\delta \to_{\beta_f} \dots$ $\omega_3 = \delta((\lambda x.\delta)(xx)) \to_{\beta_f} \delta\delta \to_{\beta_f} \delta\delta \to_{\beta_f} \dots$

Problem 1: No confluence: $(\lambda x.(\lambda z.z)(xx))\delta$

Problem 2: No subject reduction with multi types [Ehr12]: $(\lambda y.yy)(xx) \to_{\beta_f} (xx)(xx)$.
No subject expansion with multi types [Ehr12]: $(\lambda y.z)(xx) \to_{\beta_f} z$.

Solution to Problems 1–2: Just modify the syntax of the FC (a bit tricky, omitted).

Second alternative CbV λ-calculus: Value Substitution Calculus [AccPao12]

Terms: $s, t ::= v \mid ts \mid t[s/x]$ Values: $v ::= x \mid \lambda x.t$

Substitution contexts: $L ::= [t_1/x_1] \ldots [t_n/x_n]$

Reductions: $(\lambda x.t)Ls \to_m t[s/x]L$ $t[vL/x] \to_e t\{v/x\}L$

Second alternative CbV λ-calculus: Value Substitution Calculus [AccPao12]

Terms: $s, t ::= v \mid ts \mid t[s/x]$ Values: $v ::= x \mid \lambda x.t$

Substitution contexts: $L ::= [t_1/x_1] \ldots [t_n/x_n]$

Reductions: $(\lambda x.t)Ls \rightarrow_m t[s/x]L$ $\quad t[vL/x] \rightarrow_e t\{v/x\}L$

1. β_v-reduction can be simulated in the Value Substitution Calculus (VSC).

$$(\lambda x.t)v \rightarrow_m t[v/x] \rightarrow_e t\{v/x\}$$

2. VSC extends β_v-reduction: ω_1 and ω_3 are β_v-normal but

$$\omega_1 = (\lambda x.\delta)(xx)\delta \rightarrow_m \delta[xx/x]\delta \rightarrow_m (zz)[\delta/z][xx/x] \rightarrow_e \delta\delta\,[xx/x] \rightarrow \ldots$$
$$\omega_3 = \delta((\lambda x.\delta)(xx)) \rightarrow_m \delta(\delta[xx/x]) \rightarrow_m (zz)[\delta[xx/x]/z] \rightarrow_e \delta\delta\,[xx/x] \rightarrow \ldots$$

Second alternative CbV λ-calculus: Value Substitution Calculus [AccPao12]

Terms: $s, t ::= v \mid ts \mid t[s/x]$ Values: $v ::= x \mid \lambda x.t$

Substitution contexts: $L ::= [t_1/x_1] \ldots [t_n/x_n]$

Reductions: $(\lambda x.t)Ls \to_m t[s/x]L$ $t[vL/x] \to_e t\{v/x\}L$

1. β_v-reduction can be simulated in the Value Substitution Calculus (VSC).

$$(\lambda x.t)v \to_m t[v/x] \to_e t\{v/x\}$$

2. VSC extends β_v-reduction: ω_1 and ω_3 are β_v-normal but

$$\omega_1 = (\lambda x.\delta)(xx)\delta \to_m \delta[xx/x]\delta \to_m (zz)[\delta/z][xx/x] \to_e \delta\delta\,[xx/x] \to \ldots$$
$$\omega_3 = \delta((\lambda x.\delta)(xx)) \to_m \delta(\delta[xx/x]) \to_m (zz)[\delta[xx/x]/z] \to_e \delta\delta\,[xx/x] \to \ldots$$

Theorem (Subject reduction and expansion, [AccGue18]): If $t \to_{VSC} u$ then $[\![t]\!]_{\vec{x}} = [\![u]\!]_{\vec{x}}$.

$$\frac{\Gamma, x : M \vdash t : N \quad \Delta \vdash s : M}{\Gamma + \Delta \vdash t[s/x] : N}\, ES$$

Termination equivalence: weak but not strong

Consider weak reduction (i.e. not firing redexes under λ's): perfect match!

Prop (Diamond) Both VSC-reduction and β_f-reduction are diamond.

Thm (Termination equivalence [AccGue16]): t is VSC-normalizing iff t is β_f-normalizing.

Thm (Correctness & completeness [AccGue18]): t is VSC-normalizing iff $[\![t]\!]_{\vec{x}} \neq \emptyset$.

Termination equivalence: weak but not strong

Consider weak reduction (i.e. not firing redexes under λ's): perfect match!

Prop (Diamond) Both VSC-reduction and β_f-reduction are diamond.

Thm (Termination equivalence [AccGue16]): t is VSC-normalizing iff t is β_f-normalizing.

Thm (Correctness & completeness [AccGue18]): t is VSC-normalizing iff $[\![t]\!]_{\vec{x}} \neq \emptyset$.

With strong reduction (i.e. firing redexes everywhere): A mess! (and the diamond fails)

Problem 1: VSC-reduction and β_f-reduction have different notions of termination.

Problem 2: Characterization of VSC-normalization or β_f-normalization with multi types?

Table of Contents

The external strategy

In CbN, the leftmost-outermost strategy (LO) fires the leftmost-outermost β-redex.

Thm (Normalization [CurFey58]): If t is β-normalizing then LO from t terminates.

The external strategy

In CbN, the leftmost-outermost strategy (LO) fires the leftmost-outermost β-redex.

Thm (Normalization [CurFey58]): If t is β-normalizing then LO from t terminates.

What is the analog for Strong CbV? Things are a bit trickier!

Def: The external strategy (roughly) fires a redex everywhere, except under λ's in a *irrelevant* position for normalization (e.g. an applied λ or a λ on the right of another λ).

Rmk: The external strategy $\rightarrow_x$ is not deterministic but diamond.

Ex: If $I = \lambda z.z$, $\rightarrow_x$ cannot fire the redex II in $(\lambda x.x(II))v$. But $\lambda x.\delta\delta \rightarrow_x \lambda x.\delta\delta \rightarrow_x \ldots$

The external strategy

In CbN, the leftmost-outermost strategy (LO) fires the leftmost-outermost β-redex.

Thm (Normalization [CurFey58]): If t is β-normalizing then LO from t terminates.

What is the analog for Strong CbV? Things are a bit trickier!

Def: The external strategy (roughly) fires a redex everywhere, except under λ's in a *irrelevant* position for normalization (e.g. an applied λ or a λ on the right of another λ).

Rmk: The external strategy $\rightarrow_x$ is not deterministic but diamond.

Ex: If $I = \lambda z.z$, $\rightarrow_x$ cannot fire the redex II in $(\lambda x.x(II))v$. But $\lambda x.\delta\delta \rightarrow_x \lambda x.\delta\delta \rightarrow_x \ldots$

Rmk: The external strategy behaves differently in VSC and FC.

External in FC: $t = (\lambda x.I)(y(\lambda z.\delta\delta)) \rightarrow_{x\beta_f} I$ (which is normal)

External in VSC: $t \rightarrow_{xVSC} I[y(\lambda z.\delta\delta)/x] \rightarrow^*_{xVSC} I[y(\lambda z.\delta\delta)/x] \rightarrow_{xVSC} \cdots$

More about the external strategy

Question: Why is the external strategy important?

[AccConSac21,BieChaDra20] proved that it is a reasonable cost model for Strong CbV.

More about the external strategy

Question: Why is the external strategy important?

[AccConSac21,BieChaDra20] proved that it is a reasonable cost model for Strong CbV.

Def: In VSC, the external reduction $\to_{xVSC}$ is formally defined as follows:

Rigid terms:	$r ::= x \mid r\,t \mid r[r'/x]$
Rigid contexts:	$R ::= rX \mid Rt \mid R[r/x] \mid r[R/x]$
External contexts:	$X ::= \langle\cdot\rangle \mid \lambda x.X \mid t[R/x] \mid X[r/x] \mid R$

$\to_{xVSC}$ is the closure under external contexts of weak (i.e. not under λ) reduction in VSC.

More about the external strategy

Question: Why is the external strategy important?

[AccConSac21,BieChaDra20] proved that it is a reasonable cost model for Strong CbV.

Def: In VSC, the external reduction $\to_{xVSC}$ is formally defined as follows:

Rigid terms:	$r ::= x \mid r\,t \mid r[r'/x]$
Rigid contexts:	$R ::= rX \mid Rt \mid R[r/x] \mid r[R/x]$
External contexts:	$X ::= \langle\cdot\rangle \mid \lambda x.X \mid t[R/x] \mid X[r/x] \mid R$

$\to_{xVSC}$ is the closure under external contexts of weak (i.e. not under λ) reduction in VSC.

Rmk: In VSC, t is normal for $\to_{xVSC}$ iff t is normal for $\to_{VSC}$.

Rmk: Rigid terms are not λ's, have a free head variable, their unfolding is still rigid.

Multi types for Strong CbV

Goal: We want to prove that the external strategy is normalizing for Strong CbV.

Questions: For which Strong CbV? And how to prove it?

Multi types for Strong CbV

Goal: We want to prove that the external strategy is normalizing for Strong CbV.

Questions: For which Strong CbV? And how to prove it?

Idea: Let's use multi types, the same type system as in [Ehr12]! But it's trickier!

Ex: $\lambda x.\delta\delta$ is external divergent but typable with $[\,]$ (use the rule ! with no premises).

Ex: $y\lambda x.\delta\delta$ is external divergent but typable with $y : [[\,] \multimap M] \vdash y\lambda x.\delta\delta : M$, for some M.

Multi types for Strong CbV

Goal: We want to prove that the external strategy is normalizing for Strong CbV.

Questions: For which Strong CbV? And how to prove it?

Idea: Let's use multi types, the same type system as in [Ehr12]! But it's trickier!

Ex: $\lambda x.\delta\delta$ is external divergent but typable with [] (use the rule ! with no premises).

Ex: $y\lambda x.\delta\delta$ is external divergent but typable with $y : [[\,] \multimap M] \vdash y\lambda x.\delta\delta : M$, for some M.

Idea: Let's forbid [] on the right of $\vdash$ and on the left of arrow types in the environment

- $\rightsquigarrow$ no [] in the positive positions on the right of $\vdash$ (right shrinking types);
- $\rightsquigarrow$ dually, no [] in the negative positions on the left of $\vdash$ (left shrinking types).

Shrinking types, formally

We take the same type system as [Ehr12], we just restrict the types.

Right multi shrink. $M^r ::= [L^r_1, \ldots, L^r_n]$ $(n \geq 1)$ Right linear shrink. $L^r ::= * \mid M^\ell \multimap M^r$

Left multi shrink. $M^\ell ::= [L^\ell_1, \ldots, L^\ell_n]$ $(n \geq 0)$ Left linear shrink. $L^\ell ::= * \mid M^r \multimap M^\ell$

Shrinking types, formally

We take the same type system as [Ehr12], we just restrict the types.

Right multi shrink. $M^r ::= [L^r_1, \ldots, L^r_n]$ $(n \geq 1)$ Right linear shrink. $L^r ::= * \mid M^\ell \multimap M^r$

Left multi shrink. $M^\ell ::= [L^\ell_1, \ldots, L^\ell_n]$ $(n \geq 0)$ Left linear shrink. $L^\ell ::= * \mid M^r \multimap M^\ell$

Def: An environment $x_1 : M_1, \ldots, x_n : M_n$ is left shrinking if all M_i's are left shrinking.
A typing $(\Gamma; M)$ is shrinking if Γ is left shrinking and M is right shrinking.

Shrinking types, formally

We take the same type system as [Ehr12], we just restrict the types.

Right multi shrink. $M^r ::= [L^r_1, \dots, L^r_n]$ $(n \geq 1)$ Right linear shrink. $L^r ::= * \mid M^\ell \multimap M^r$

Left multi shrink. $M^\ell ::= [L^\ell_1, \dots, L^\ell_n]$ $(n \geq 0)$ Left linear shrink. $L^\ell ::= * \mid M^r \multimap M^\ell$

Def: An environment $x_1 : M_1, \dots, x_n : M_n$ is left shrinking if all M_i's are left shrinking.
A typing $(\Gamma; M)$ is shrinking if Γ is left shrinking and M is right shrinking.

Lemma (Shrinking spread) Let $\Pi \triangleright \Gamma \vdash r : M$, r rigid. If Γ is left shrinking then so is M.

The key results 1: Shrinking typability $\Rightarrow$ external normalization

Thm (Quantitative subject reduction) Let $\Pi \triangleright \Gamma \vdash t : M$ a derivation where (Γ, M) is shrinking. If $t \to_{xVSC} t'$ then there is a derivation $\Pi' \triangleright \Gamma \vdash t' : M$ with $|\Pi| > |\Pi'|$.

The key results 1: Shrinking typability $\Rightarrow$ external normalization

Thm (Quantitative subject reduction) Let $\Pi \triangleright \Gamma \vdash t : M$ a derivation where (Γ, M) is shrinking. If $t \to_{xVSC} t'$ then there is a derivation $\Pi' \triangleright \Gamma \vdash t' : M$ with $|\Pi| > |\Pi'|$.

Rmk: Dropping shrinkingness, the quantitative aspect is false! $\lambda x.\delta\delta \to_{xVSC} \lambda x.(zz)[\delta/z]$ but both terms are only typable with $[\,]$ using the ! rule with no premises $\rightsquigarrow$ $|\Pi| = |\Pi'|$.

The key results 1: Shrinking typability $\Rightarrow$ external normalization

Thm (Quantitative subject reduction) Let $\Pi \triangleright \Gamma \vdash t : M$ a derivation where (Γ, M) is shrinking. If $t \rightarrow_{xVSC} t'$ then there is a derivation $\Pi' \triangleright \Gamma \vdash t' : M$ with $|\Pi| > |\Pi'|$.

Rmk: Dropping shrinkingness, the quantitative aspect is false! $\lambda x.\delta\delta \rightarrow_{xVSC} \lambda x.(zz)[\delta/z]$ but both terms are only typable with $[\,]$ using the ! rule with no premises $\rightsquigarrow$ $|\Pi| = |\Pi'|$.

Rmk: Replacing $\rightarrow_{xVSC}$ with $\rightarrow_{VSC}$, quantitativity fails! $I(\lambda x.II) \overset{\not\rightarrow_{xVSC}}{\rightarrow_{VSC}} I(\lambda x.z[I/z])$ and

$$\Pi = \cfrac{\cfrac{\cfrac{\cfrac{}{\vdash y : [\,]}\,!}{\vdash \lambda y.y : [\,] \multimap [\,]}\,\lambda}{\vdash \lambda y.y : [[\,] \multimap [\,]]}\,! \qquad \cfrac{}{\vdash \lambda x.II : [\,]}\,!}{\vdash I(\lambda x.II) : [\,]}\,@$$

but any $\Pi' \triangleright \vdash I(\lambda x.z[I/z]) : [\,]$ is s.t. $|\Pi'| \geq |\Pi|$.

Thm (Shrinking correctness) Let $\Pi \triangleright \Gamma \vdash t : M$ a derivation where (Γ, M) is shrinking. Then $t \rightarrow^*_{xVSC} u$ where u is VSC-normal.

The key results 2: External normalization $\Rightarrow$ shrinking typability

Lemma: Every VSC-normal form is typable with a shrinking typing.

Thm (Shrinking completeness) Let $t \to^*_{xVSC} u$ where u is VSC-normal. Then $\Pi \triangleright \Gamma \vdash t : M$ a derivation where $(\Gamma; M)$ is shrinking.

The key results 2: External normalization ⇒ shrinking typability

Lemma: Every VSC-normal form is typable with a shrinking typing.

Thm (Shrinking completeness) Let $t \to^*_{xVSC} u$ where u is VSC-normal. Then $\Pi \triangleright \Gamma \vdash t : M$ a derivation where $(\Gamma; M)$ is shrinking.

Cor: If t is VSC-normalizing then t is VSC-normalizing with the external strategy.

The key results 2: External normalization $\Rightarrow$ shrinking typability

Lemma: Every VSC-normal form is typable with a shrinking typing.

Thm (Shrinking completeness) Let $t \to^*_{xVSC} u$ where u is VSC-normal. Then $\Pi \triangleright \Gamma \vdash t : M$ a derivation where $(\Gamma; M)$ is shrinking.

Cor: If t is VSC-normalizing then t is VSC-normalizing with the external strategy.

Shrinking types define a denotational model: shrinking relational semantics:

$$[\![t]\!]^{\mathsf{shr}}_{\vec{x}} = \{(\Gamma, M) \text{ shrinking} \mid \Gamma \vdash t : M \text{ is derivable}\} \qquad \text{where } \vec{x} \subseteq \mathsf{fv}(t)$$

which is adequate: $[\![t]\!]^{\mathsf{shr}}_{\vec{x}} \neq \emptyset$ iff t is VSC-normalizing.

The key results 2: External normalization ⇒ shrinking typability

Lemma: Every VSC-normal form is typable with a shrinking typing.

Thm (Shrinking completeness) Let $t \to^*_{xVSC} u$ where u is VSC-normal. Then $\Pi \triangleright \Gamma \vdash t : M$ a derivation where $(\Gamma; M)$ is shrinking.

Cor: If t is VSC-normalizing then t is VSC-normalizing with the external strategy.

Shrinking types define a denotational model: shrinking relational semantics:

$$[\![t]\!]^{\mathsf{shr}}_{\vec{x}} = \{(\Gamma, M) \text{ shrinking} \mid \Gamma \vdash t : M \text{ is derivable}\} \qquad \text{where } \vec{x} \subseteq \mathsf{fv}(t)$$

which is adequate: $[\![t]\!]^{\mathsf{shr}}_{\vec{x}} \neq \emptyset$ iff t is VSC-normalizing.

Rmk: Shrinking completeness fails in FC, see counterexample on p. 15: $(\lambda x.I)(y(\lambda z.\delta\delta))$.
$\rightsquigarrow$ The shrinking relational semantics suggests that VSC is the “right” Strong CbV.

Thank you!

Questions?